

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR U.S. LETTERS PATENT

Title:

**MEMORY ACCESS METHOD BY REFERENCING DATA,  
AND DATA PROCESSING DEVICE USING THE SAME**

Inventors:

Motoyuki KATO  
Hiroyuki YANAGI  
Shinji NAKAGAWA  
Yosuke BABA

Dickstein Shapiro Morin  
& Oshinsky LLP  
2101 L Street NW  
Washington, DC 20037-1526

T057350 = 44620650

# MEMORY ACCESS METHOD BY REFERENCING DATA, AND DATA PROCESSING DEVICE USING THE SAME

## **Field Of The Invention**

5

This invention concerns a scheme to execute a program through the use of an interpreter language and an information processing device which contains such a program. More specifically, this invention concerns a technique to execute a program which requires that destinations to be accessed in the memory be specified by referencing data.

19



## **Background Of The Invention**

Java®, the object-oriented programming language developed by Sun Microsystems, operates in an environment which has a Java interpreter to convert Java commands into byte code without having to rely on a platform. It has received a great deal of attention as a language which is well-suited to set values in information processing systems contained in various devices.

20 To prevent improper access to the memory, information processing systems using Java do not allow the program to directly specify a destination to be accessed in the memory, but instead specify the destination indirectly through data representing its name.

For example, a program indicating that a value should be set in a variable (i.e., in a field) or that this value be read out would designate the variable by the name of the class to which it belongs, the name of the variable and the name of the type of variable (e.g., int or byte). Programs that access methods designate the method they wish to access by the name of the class to which the method belongs and the name of the method. Generally, the processing entailed in using a given convention (such as character data indicating a name) to find the location where data are stored concerning something to be processed (a variable, a method, a

class, etc.) is called "referencing." Indicating that data are to be referenced is called a "reference request." (For example, referencing a variable, a method or a class is called a "field reference," a "method reference" or a "class reference.")

5 When executing a program which requests a reference using a given name as data, the Java interpreter which is called a "virtual machine" will retrieve the specified reference table (as will be explained in detail shortly) by means of the data represented by the name and specify the address where the data to be referenced are stored. Referencing data designated in this way and indicating the location where they are stored is called "resolving a reference."

10

Figure 6 shows the compiled result of a program which uses Java. In the figure, each dotted rectangle is one unit's worth of program code. Each code contains multiple data points. In the example, individual data are shown in the white boxes.

15

In Figure 6, 10 is the byte code program obtained by converting the message indicating that a value should be assigned to the variable. (In assembler code this is represented as "putstatic Sample.value:int".) It consists of instruction code representing the content of the request (B3) and an operand representing the address (00 06) where the content which the instruction indicated was to be referenced can be found. (Hereafter, an object program with 20 an instruction code attached will be referred to as an "instruction.")

25

Reference data configured as shown on the right in the figure are linked to instruction 10. These data consist of a combination of various lengths of code data, to the head of which is attached a tag representing the type of data. An identification number called the "constant pool entry number" (hereafter, called simply the "entry number") is attached to each set of code data, and a link is set up for each entry number. The first entry number for reference data, 06, is written into the aforesaid operand. By following the links between the various sets of code data in order, starting from the first entry, we can obtain a character string representing the name of the class, the variable, and the type of variable that were described to the 30 source program.

These reference data are stored in class units. (Hereafter, reference data in class units are referred to as "class file data.") To simplify the explanation, in Figure 6 the reference data which have been selected are for a single instruction. Entry numbers not shown in the figure  
5 are used for the reference data for other instructions.

In the figure, character data representing the class name "Sample" are recorded in entry number 20; data representing the variable name "Value" are recorded in entry number 27.

The character string "I", which represents int, the name of the type of variable, is recorded in  
10 entry number 17. The character strings of data in entry numbers 20, 27 and 17 are encoded. Code data representing the number of bytes in the character string and the tag "01", which indicates that character strings of encoded data are recorded, are added in front of the encoded data.

The data in entry numbers 1, 6 and 9 indicate what location the aforesaid character string is to reference.

The data for entry number 6 have the tag "09", which means "field reference". This tag indicates that the following data are field reference data. Code data representing the entry  
20 numbers 1 and 9 are recorded in entry number 6.

The tag "07", which means "class reference", is attached to entry number 1. This sets up a link in the character data which represent the aforesaid class name to connect this entry number with entry number 20, the destination to be referenced. The tag "0C", which means  
25 "name and type reference", is attached to entry number 9. This links entry number 9 to entry number 27, where the character data for the aforesaid variable name are stored, and entry number 17, where the name of the type of variable is stored.

This configuration of data allows us to follow the links in the reference data in an orderly fashion beginning with entry number 6, which is written into the operand of instruction  
30

10. In this way we can obtain the names of the class, the variable and the type, which we need for a field reference. Other commands express the data in the same fashion. When the source program is compiled, instructions are generated which serve as links to the reference data that represent the actual content of the designated reference.

5

Reference data can be shared among a number of instructions which all require the same reference. For example, let us assume that the aforesaid instruction 10 occurs in a number of places in the program, and that there is another instruction (for example, an instruction obtained from the program "getstatic Sample.Value:int") which requires the same field reference as instruction 10. The aforesaid entry number 6 is then written into the operand of each of these instructions to set up a link to the reference data in the aforesaid Figure 6.

10

5

When the interpreter executes each instruction, it recognizes the content of the command from the aforesaid instruction. It follows the links in order starting from the entry number written into the operand, and it finds the content of the designated reference and the data which represent the name used in the reference.

For the different variables and methods set up in each class, a reference table is created in the system to specify from the name data the location in the memory where those variables and programs are stored. The interpreter uses the name data it obtains from the aforesaid reference data to look up the aforesaid reference table. It executes the necessary processing to find the destination designated by the aforesaid instruction.

Figure 7 shows the configuration of the table of classes and the table of fields (11 and 25 12 in the drawing) used to find field references. In Figure 7, 13 is a heap memory which serves as the work area for all the objects designated by the program. 14 is a table where character strings such as names of classes or variables are stored.

As the aforesaid class file data are generated, data concerning the class are stored in the 30 aforesaid tables 11 and 12. Data stored in class table 11 include the top address of the class;

the top address of the string; the name index for the class itself; the name index for the parent class (the super index); the size of objects in the class; and access flags. (Hereafter, these data are all called "class data".)

5        The top address of a class is the address where the head data of the aforesaid class file are stored. The top address of the string is the head address of the area in table 14 where character strings such as the names of classes and variables for each class are stored. The name index consists of the constant pool entry numbers which indicate the names of the classes in the aforesaid reference data. Access flags indicate whether or not a class can be accessed.

10

An individual index (hereafter referred to as a "class table index") is attached to the class data for each class.

15

For every variable the program creates, various data are stored in the field table, including an object offset; an index to the class table; a name index; a type index; and access flags. (Hereafter, these data are collectively referred to as "field data".)

20

The aforesaid object offset is a value indicating the relative location of the field assigned to the aforesaid variable in the work area for the object which contains that variable. This value is determined when the class file data are created. The size of the work area is determined in which the instance of each class must be stored in a heap memory, and a storage area is allotted for each variable in this work area.

However, values for static variables are themselves stored in the field table.

25

The index to a class table is attached to the class data in the aforesaid class table for the class which contains the aforesaid variable. Name and type indices are constant pool entry numbers for the data which represent variable and type names in the aforesaid reference data. Access flags, as was just as was described above, indicate whether or not a variable can be accessed.

30

5 A separate index (Hereafter referred to as a "field table index") is attached to each set of file data.

10 To find a field reference, the interpreter uses the name of the class it obtains from the reference data linked to the aforesaid instruction to retrieve the class table and get the table index for that class. It then retrieves the field table and extracts from the data containing the aforesaid index the field data whose variable and type names match the names it got from the aforesaid reference data. It gets the address of the storage area assigned to the variable from the object offset value in the field data.

15 If the interpreter is to execute the same instruction again, when it obtains the result of referencing the aforesaid field, it rewrites the instruction code from the aforesaid instruction into the code which it has found. It rewrites the operand into the number of the field table index it obtained from the aforesaid field reference. As a result of this rewriting, when the interpreter executes the instruction, it can immediately read the object offset that is in the field table from the field table index written in the operand and thereby obtain the location where the designated variable is stored.

20 To aid in referencing methods, a reference table (the method table) is set up to direct the interpreter, via class and method names, to the locations where programs for methods are stored. The interpreter obtains the index for the method table from the instruction and from the name of the method it reads out of the reference data linked to the instruction, and in this way it references the method. If it is to execute the instruction again, the interpreter rewrites its operand in the aforesaid index for the method table which it has obtained.

25 Figure 8 shows the chain of processing in a system using a byte code program generated by the aforesaid Java source program, from the time the system is started until the program is completed.

The object program and class file data constituting this system are stored in an internal ROM. When power is supplied to start the system, the object program is loaded into a RAM, and the class which is to be activated first by the start-up program is indicated (Step 1). In Step 2, the class file data for the indicated class are read out into the RAM. In Step 3, the data 5 for the aforesaid class whose file data have been read out are stored in various reference tables, including the class table, the field table and the method table.

In Step 4, when the environment for executing an instruction has been arranged in this way, the first instruction is read. If this instruction requires that a class be referenced for which 10 class file data have not yet been created, we move from Step 5 to Step 6 and designate that file data be created for this class. In response, file data are created and entered into the reference tables in Steps 2 and 3. When this has been completed, the aforesaid instruction is read again.

If file data already exist for the class whose reference is required by the instruction, the 15 interpreter finds the names of the class, variable and method in the file data. If this instruction is the first one executed after the system is started up, its instruction code will not have been referenced yet, so we proceed from Step 7 to Step 8.

In Step 8, the interpreter follows the link in the file data from the entry number written 20 into the operand of the aforesaid instruction and finds the names of the class and the variable (or the method). From the names it has found, it references the various tables and finds the references which specify the locations where the variable and the method program are stored. In Step 9, the interpreter accesses the addresses it obtained by finding the aforesaid references. When it has completed the instruction, it goes from Step 10 back to Step 4. Thereafter, the 25 same processing is executed for each instruction.

When the reference processing is completed in Step 8, the instruction code and operand in the completed instruction are rewritten. When this rewritten instruction is read, the interpreter proceeds from Step 7 to Step 9 and executes the aforesaid instruction without 30 having to find the references.

If we follow the procedure described above, when the first instruction is executed after the system is started, all instructions requiring references must execute the processing in Step 8. In this processing, as was discussed earlier, several stages of referencing must be pursued to 5 arrive at the storage location of the variable or method. This causes a considerable delay between the time the instruction is read and the time it is executed. This causes the processing to be extremely slow when a program is executed immediately after the system is started. Further, since a program's running time immediately after the system is started will be different from the same program's running time the second time or third time it is run, when its 10 references have already been found, it will prove difficult to use such a program in a device for real-time processing which requires an estimated running time.

In prior art systems, even if there were instructions required the same reference in a number of places in a continuous program, that reference had to be found anew for each instruction. This made the referencing efficiency extremely poor.

If the instruction code and operand are rewritten, the object program must be stored in the RAM. It will then take time to load the program in the RAM when the device is started, which will increase the start-up time that the system requires. A RAM with the same capacity 20 as the ROM must be available to store the object program, which drives up the cost of the device.

### Summary Of The Invention

25 This invention was developed in consideration of the problems described above. Its first objective is to realize an information processing scheme such that the necessary references, including reference data specifying locations to be accessed in the memory, are found before the program is executed, and each result is stored in a link to the program. Then the locations to be accessed can be specified immediately by the reference data when the program is exe-

cuted. The program's running time is stabilized and its speed is increased. It is well suited for use in a device which requires real time capability.

5 The second objective of this invention is to eliminate the need to load the program into a RAM. This would shorten the start-up time for the system, greatly reduce the required capacity of the RAM, and lower the production cost of the device.

10 The third objective of this invention is to enable the reference results to be read out of the ROM along with the program before the program is executed. This would further reduce the required capacity of the RAM and, when this system is used in a device, enable the program to run at high speed immediately after the system is started up.

15 With the invention disclosed in Claim 1 of this application, when a program written in an interpreter language is run, the reference data specifying the locations to be accessed in the memory are extracted and the references are found before the program is executed. The results which are obtained are then linked to the program through the aforesaid reference data. When a program is executed which requires that certain data be referenced and accessed in the memory, the locations to be accessed in the aforesaid memory are specified based on the results linked to the program through these reference data.

20 The invention disclosed in Claim 2 of this application comprises an information processing device which contains a program written in an interpreter language. To enable it to implement the scheme outlined above, it has a means to execute the aforesaid program and a means to link results of referencing to the program through the reference data specifying 25 which locations are to be accessed in the memory. When it is to execute a program requiring that specified data be referenced and accessed in the memory, the means to execute the program uses the results of referencing, now linked to the aforesaid program through the reference data, to specify the locations to be accessed in the aforesaid memory.

If the aforesaid program in an interpreter language is a byte code program compiled from a source program written in Java, it consists of an object program (i.e., an instruction) requiring that names of variables and methods be referenced and data representing the actual contents (the names of the class, the variable, the method, etc.) of the reference data linked to 5 the instruction. The means to execute the program is realized by the function of the interpreter. Through the aforesaid linked data, it verifies the content of the reference data as it executes programs in a given order.

Using the aforesaid reference data to find a reference means referencing certain data by 10 means of the reference data and specifying the locations to be accessed in the memory so that no further reference processing will be necessary.

For example, finding a field reference would mean using the character data indicated by the program, such as the name of the class and variable, to obtain the address where the specified variable is stored and the data which correspond to that address (the aforesaid index to the 15 field table). Finding a method reference would mean using character data such as the name of the method to obtain the address where the specified method is stored and the data which correspond to that address (the aforesaid index to the method table). Finding a class reference would mean using character data representing the name of the class to obtain the address 20 where the data related to the methods and variables in the indicated class (the aforesaid class data) are stored and the data which correspond to that address (the aforesaid index to the class table).

With the invention disclosed in Claim 3 of this application, if the aforesaid program 25 consists of an object program in byte code and data which represent the content of reference data linked to that program, the results of the aforesaid referencing will be stored in the link information of the aforesaid object program. With the invention disclosed in Claim 4 of this application, the link information contains code data of a number of fixed lengths. The aforesaid results of referencing are stored in a location determined by the head code data.

With the invention disclosed in Claim 5 of this invention, the object program in byte code and its link information are read out of the ROM to execute the program.

With the invention disclosed in Claim 1 of this application, the results obtained by referencing the data which specify locations to be accessed in the memory are stored as links to the program before it is executed. This allows the memory to be accessed speedily and the designated processing to be executed immediately after the system is activated. It also stabilizes the program's running time, making it possible to accurately predict that time.

10 The results of referencing the data are linked to the program through the reference data. This makes it unnecessary to rewrite the program as the data are referenced or load the program into the RAM when the system is activated.

15 The fact that the results of referencing are linked to the program without rewriting it means that a program which requires the same reference data to be looked up more than once will not have to repeatedly reference them as was the case in prior art programs.

20 With the invention disclosed in Claim 2 of this application, for every set of reference data the previously obtained results of referencing will be linked to the program at the moment the system is activated. This enables processing to be executed speedily as soon as the program is started up.

25 With the invention disclosed in Claim 3 of this application, the results of referencing the data are stored in the link information indicating the content of the reference data specified by the object program. In this way the results of referencing are linked to the program using the existing configuration of the program's links.

30 With the invention disclosed in Claim 4 of this application, the results of referencing the data are stored in a location specified by the head code data in the link information. This enables the results to be retrieved easily. The location where the referencing results for each

data file are stored is checked before the program is run. This enables the referencing results to be distinguished easily.

With the invention disclosed in Claim 5 of this application, the link information containing the referencing results is stored with the object program in a ROM. This allows the program to run at high speed immediately after the system is started up.

#### Brief Description Of The Drawings

10

Figure 1 shows the configuration of an information processing system related to the first preferred embodiment of this invention.

15

Figure 2 illustrates how the class file data are configured in this embodiment.

20

Figure 3 shows the order of processing in an information processing system configured as in Figure 1 from the moment the system is started up until the program has been executed.

25

Figure 4 shows details of the processing in Step 4 of the aforesaid Figure 3.

30

Figure 5 shows the configuration of an information processing system related to the second preferred embodiment of this invention.

25

Figure 6 illustrates how the class file data are configured according to a prior art.

30

Figure 7 shows the configuration of the table of classes and the table of fields used to find field references.

Figure 8 shows the chain of processing in a system according to the prior art.

### Detailed Description Of The Invention

Figure 1 shows the configuration of an information processing system related to the  
5 first preferred embodiment of this invention.

This information processing system comprises a device which contains a computer. The  
system executes a byte code program (hereafter simply referred to as "a program") compiled  
from a Java source program. It consists of storage unit 1 for instructions; storage unit 2 for  
10 class file data; unit 3 to execute instructions; unit 4 to manage tables; unit 5 to resolve refer-  
ences; heap memory 6; and unit 7 to assign memory. Of these components, the four process-  
ing units, execution unit 3, management unit 4, resolution unit 5 and assignment unit 7, are  
realized by a Java interpreter. Instruction storage unit 1 is set up in the computer's ROM;  
data storage unit 2, heap memory 6 and the various reference tables created by management  
45 unit 4 are set up in a RAM.

Among the byte code programs, only the aforesaid instructions, i.e., the object program  
consisting of instruction codes and operands, are stored in the aforesaid instruction storage  
unit 1.

20 Class file data consisting of the reference data assembled for each class which indicate  
the actual content of the references specified by instructions are set up in storage unit 2. The  
reference data in a given class are represented by numerous data sets linked to the program  
using the aforesaid entry numbers. The head entry number of the reference data specified by  
25 an instruction which requests a reference is written into the operand of that instruction.

Table management unit 4 creates reference tables, such as a class table, a field table and  
a method table, from the class file data for each class just as was done in prior art schemes.  
The data organized in these reference tables are supplied as needed to execution unit 3, reso-  
30 lution unit 5 and assignment unit 7.

Execution unit 3 reads the instructions out of storage unit 1 one by one and executes them.

5 Memory assignment unit 7 sets up a work area in heap memory 6 for each instance generated by the execution of the program in response to a command from execution unit 3. Based on the value of the aforesaid object offset, it assigns a given address in the said work area to the variables for each instance.

10 Resolution unit 5 resolves references requested by an instruction. In this embodiment, references indicated by name data contained in the class file are resolved by finding the names before the program is executed, and the results obtained are written into the class file data.

15 The technique used to resolve the references is identical to that employed in the prior art. For example, the result of resolving a field reference would be a field table index, and the result of resolving a method reference would be a method table index. These indices would be written into the class file data. For a class reference, a class table index would be obtained for the class name that was indicated and stored as a result in the class file data.

20 Figure 2 illustrates how the class file data are configured in this embodiment. Just as in the prior art configuration shown in Figure 6, the figure extracts the reference data for a single instruction.

25 In this embodiment, as in the prior art, various code data are linked to the code data for entry number 6, the "field reference" which is at the head of the list. These character data indicate the name of the class, variable and type of variable, as well as what sort of reference they represent.

These code data have a fixed length. As in the prior art, a flag is stored at the head of each set of code data. Other data are stored at the end of the code (in the example, on the right side). A blank space (indicated in the drawing by a dash) is left in the center.

5 Because this embodiment requires that all code data must have a fixed length, the character string representing the name is stored in a separate table of character strings. The aforesaid table is organized in class units. The table number for the character string table (in the drawing, 0) and the locations of those character strings in the table (in the drawing, represented by the numerals 1000, 1004 and 1010) are stored as location data in entry numbers 17,  
10 20 and 27.

1000  
1004  
1010  
1014  
1018  
1022  
1026  
1030  
1034  
1038  
1042  
1046  
1050  
1054  
1058  
1062  
1066  
1070  
1074  
1078  
1082  
1086  
1090  
1094  
1098  
1102  
1106  
1110  
1114  
1118  
1122  
1126  
1130  
1134  
1138  
1142  
1146  
1150  
1154  
1158  
1162  
1166  
1170  
1174  
1178  
1182  
1186  
1190  
1194  
1198  
1202  
1206  
1210  
1214  
1218  
1222  
1226  
1230  
1234  
1238  
1242  
1246  
1250  
1254  
1258  
1262  
1266  
1270  
1274  
1278  
1282  
1286  
1290  
1294  
1298  
1302  
1306  
1310  
1314  
1318  
1322  
1326  
1330  
1334  
1338  
1342  
1346  
1350  
1354  
1358  
1362  
1366  
1370  
1374  
1378  
1382  
1386  
1390  
1394  
1398  
1402  
1406  
1410  
1414  
1418  
1422  
1426  
1430  
1434  
1438  
1442  
1446  
1450  
1454  
1458  
1462  
1466  
1470  
1474  
1478  
1482  
1486  
1490  
1494  
1498  
1502  
1506  
1510  
1514  
1518  
1522  
1526  
1530  
1534  
1538  
1542  
1546  
1550  
1554  
1558  
1562  
1566  
1570  
1574  
1578  
1582  
1586  
1590  
1594  
1598  
1602  
1606  
1610  
1614  
1618  
1622  
1626  
1630  
1634  
1638  
1642  
1646  
1650  
1654  
1658  
1662  
1666  
1670  
1674  
1678  
1682  
1686  
1690  
1694  
1698  
1702  
1706  
1710  
1714  
1718  
1722  
1726  
1730  
1734  
1738  
1742  
1746  
1750  
1754  
1758  
1762  
1766  
1770  
1774  
1778  
1782  
1786  
1790  
1794  
1798  
1802  
1806  
1810  
1814  
1818  
1822  
1826  
1830  
1834  
1838  
1842  
1846  
1850  
1854  
1858  
1862  
1866  
1870  
1874  
1878  
1882  
1886  
1890  
1894  
1898  
1902  
1906  
1910  
1914  
1918  
1922  
1926  
1930  
1934  
1938  
1942  
1946  
1950  
1954  
1958  
1962  
1966  
1970  
1974  
1978  
1982  
1986  
1990  
1994  
1998  
2002  
2006  
2010  
2014  
2018  
2022  
2026  
2030  
2034  
2038  
2042  
2046  
2050  
2054  
2058  
2062  
2066  
2070  
2074  
2078  
2082  
2086  
2090  
2094  
2098  
2102  
2106  
2110  
2114  
2118  
2122  
2126  
2130  
2134  
2138  
2142  
2146  
2150  
2154  
2158  
2162  
2166  
2170  
2174  
2178  
2182  
2186  
2190  
2194  
2198  
2202  
2206  
2210  
2214  
2218  
2222  
2226  
2230  
2234  
2238  
2242  
2246  
2250  
2254  
2258  
2262  
2266  
2270  
2274  
2278  
2282  
2286  
2290  
2294  
2298  
2302  
2306  
2310  
2314  
2318  
2322  
2326  
2330  
2334  
2338  
2342  
2346  
2350  
2354  
2358  
2362  
2366  
2370  
2374  
2378  
2382  
2386  
2390  
2394  
2398  
2402  
2406  
2410  
2414  
2418  
2422  
2426  
2430  
2434  
2438  
2442  
2446  
2450  
2454  
2458  
2462  
2466  
2470  
2474  
2478  
2482  
2486  
2490  
2494  
2498  
2502  
2506  
2510  
2514  
2518  
2522  
2526  
2530  
2534  
2538  
2542  
2546  
2550  
2554  
2558  
2562  
2566  
2570  
2574  
2578  
2582  
2586  
2590  
2594  
2598  
2602  
2606  
2610  
2614  
2618  
2622  
2626  
2630  
2634  
2638  
2642  
2646  
2650  
2654  
2658  
2662  
2666  
2670  
2674  
2678  
2682  
2686  
2690  
2694  
2698  
2702  
2706  
2710  
2714  
2718  
2722  
2726  
2730  
2734  
2738  
2742  
2746  
2750  
2754  
2758  
2762  
2766  
2770  
2774  
2778  
2782  
2786  
2790  
2794  
2798  
2802  
2806  
2810  
2814  
2818  
2822  
2826  
2830  
2834  
2838  
2842  
2846  
2850  
2854  
2858  
2862  
2866  
2870  
2874  
2878  
2882  
2886  
2890  
2894  
2898  
2902  
2906  
2910  
2914  
2918  
2922  
2926  
2930  
2934  
2938  
2942  
2946  
2950  
2954  
2958  
2962  
2966  
2970  
2974  
2978  
2982  
2986  
2990  
2994  
2998  
3002  
3006  
3010  
3014  
3018  
3022  
3026  
3030  
3034  
3038  
3042  
3046  
3050  
3054  
3058  
3062  
3066  
3070  
3074  
3078  
3082  
3086  
3090  
3094  
3098  
3102  
3106  
3110  
3114  
3118  
3122  
3126  
3130  
3134  
3138  
3142  
3146  
3150  
3154  
3158  
3162  
3166  
3170  
3174  
3178  
3182  
3186  
3190  
3194  
3198  
3202  
3206  
3210  
3214  
3218  
3222  
3226  
3230  
3234  
3238  
3242  
3246  
3250  
3254  
3258  
3262  
3266  
3270  
3274  
3278  
3282  
3286  
3290  
3294  
3298  
3302  
3306  
3310  
3314  
3318  
3322  
3326  
3330  
3334  
3338  
3342  
3346  
3350  
3354  
3358  
3362  
3366  
3370  
3374  
3378  
3382  
3386  
3390  
3394  
3398  
3402  
3406  
3410  
3414  
3418  
3422  
3426  
3430  
3434  
3438  
3442  
3446  
3450  
3454  
3458  
3462  
3466  
3470  
3474  
3478  
3482  
3486  
3490  
3494  
3498  
3502  
3506  
3510  
3514  
3518  
3522  
3526  
3530  
3534  
3538  
3542  
3546  
3550  
3554  
3558  
3562  
3566  
3570  
3574  
3578  
3582  
3586  
3590  
3594  
3598  
3602  
3606  
3610  
3614  
3618  
3622  
3626  
3630  
3634  
3638  
3642  
3646  
3650  
3654  
3658  
3662  
3666  
3670  
3674  
3678  
3682  
3686  
3690  
3694  
3698  
3702  
3706  
3710  
3714  
3718  
3722  
3726  
3730  
3734  
3738  
3742  
3746  
3750  
3754  
3758  
3762  
3766  
3770  
3774  
3778  
3782  
3786  
3790  
3794  
3798  
3802  
3806  
3810  
3814  
3818  
3822  
3826  
3830  
3834  
3838  
3842  
3846  
3850  
3854  
3858  
3862  
3866  
3870  
3874  
3878  
3882  
3886  
3890  
3894  
3898  
3902  
3906  
3910  
3914  
3918  
3922  
3926  
3930  
3934  
3938  
3942  
3946  
3950  
3954  
3958  
3962  
3966  
3970  
3974  
3978  
3982  
3986  
3990  
3994  
3998  
4002  
4006  
4010  
4014  
4018  
4022  
4026  
4030  
4034  
4038  
4042  
4046  
4050  
4054  
4058  
4062  
4066  
4070  
4074  
4078  
4082  
4086  
4090  
4094  
4098  
4102  
4106  
4110  
4114  
4118  
4122  
4126  
4130  
4134  
4138  
4142  
4146  
4150  
4154  
4158  
4162  
4166  
4170  
4174  
4178  
4182  
4186  
4190  
4194  
4198  
4202  
4206  
4210  
4214  
4218  
4222  
4226  
4230  
4234  
4238  
4242  
4246  
4250  
4254  
4258  
4262  
4266  
4270  
4274  
4278  
4282  
4286  
4290  
4294  
4298  
4302  
4306  
4310  
4314  
4318  
4322  
4326  
4330  
4334  
4338  
4342  
4346  
4350  
4354  
4358  
4362  
4366  
4370  
4374  
4378  
4382  
4386  
4390  
4394  
4398  
4402  
4406  
4410  
4414  
4418  
4422  
4426  
4430  
4434  
4438  
4442  
4446  
4450  
4454  
4458  
4462  
4466  
4470  
4474  
4478  
4482  
4486  
4490  
4494  
4498  
4502  
4506  
4510  
4514  
4518  
4522  
4526  
4530  
4534  
4538  
4542  
4546  
4550  
4554  
4558  
4562  
4566  
4570  
4574  
4578  
4582  
4586  
4590  
4594  
4598  
4602  
4606  
4610  
4614  
4618  
4622  
4626  
4630  
4634  
4638  
4642  
4646  
4650  
4654  
4658  
4662  
4666  
4670  
4674  
4678  
4682  
4686  
4690  
4694  
4698  
4702  
4706  
4710  
4714  
4718  
4722  
4726  
4730  
4734  
4738  
4742  
4746  
4750  
4754  
4758  
4762  
4766  
4770  
4774  
4778  
4782  
4786  
4790  
4794  
4798  
4802  
4806  
4810  
4814  
4818  
4822  
4826  
4830  
4834  
4838  
4842  
4846  
4850  
4854  
4858  
4862  
4866  
4870  
4874  
4878  
4882  
4886  
4890  
4894  
4898  
4902  
4906  
4910  
4914  
4918  
4922  
4926  
4930  
4934  
4938  
4942  
4946  
4950  
4954  
4958  
4962  
4966  
4970  
4974  
4978  
4982  
4986  
4990  
4994  
4998  
5002  
5006  
5010  
5014  
5018  
5022  
5026  
5030  
5034  
5038  
5042  
5046  
5050  
5054  
5058  
5062  
5066  
5070  
5074  
5078  
5082  
5086  
5090  
5094  
5098  
5102  
5106  
5110  
5114  
5118  
5122  
5126  
5130  
5134  
5138  
5142  
5146  
5150  
5154  
5158  
5162  
5166  
5170  
5174  
5178  
5182  
5186  
5190  
5194  
5198  
5202  
5206  
5210  
5214  
5218  
5222  
5226  
5230  
5234  
5238  
5242  
5246  
5250  
5254  
5258  
5262  
5266  
5270  
5274  
5278  
5282  
5286  
5290  
5294  
5298  
5302  
5306  
5310  
5314  
5318  
5322  
5326  
5330  
5334  
5338  
5342  
5346  
5350  
5354  
5358  
5362  
5366  
5370  
5374  
5378  
5382  
5386  
5390  
5394  
5398  
5402  
5406  
5410  
5414  
5418  
5422  
5426  
5430  
5434  
5438  
5442  
5446  
5450  
5454  
5458  
5462  
5466  
5470  
5474  
5478  
5482  
5486  
5490  
5494  
5498  
5502  
5506  
5510  
5514  
5518  
5522  
5526  
5530  
5534  
5538  
5542  
5546  
5550  
5554  
5558  
5562  
5566  
5570  
5574  
5578  
5582  
5586  
5590  
5594  
5598  
5602  
5606  
5610  
5614  
5618  
5622  
5626  
5630  
5634  
5638  
5642  
5646  
5650  
5654  
5658  
5662  
5666  
5670  
5674  
5678  
5682  
5686  
5690  
5694  
5698  
5702  
5706  
5710  
5714  
5718  
5722  
5726  
5730  
5734  
5738  
5742  
5746  
5750  
5754  
5758  
5762  
5766  
5770  
5774  
5778  
5782  
5786  
5790  
5794  
5798  
5802  
5806  
5810  
5814  
5818  
5822  
5826  
5830  
5834  
5838  
5842  
5846  
5850  
5854  
5858  
5862  
5866  
5870  
5874  
5878  
5882  
5886  
5890  
5894  
5898  
5902  
5906  
5910  
5914  
5918  
5922  
5926  
5930  
5934  
5938  
5942  
5946  
5950  
5954  
5958  
5962  
5966  
5970  
5974  
5978  
5982  
5986  
5990  
5994  
5998  
6002  
6006  
6010  
6014  
6018  
6022  
6026  
6030  
6034  
6038  
6042  
6046  
6050  
6054  
6058  
6062  
6066  
6070  
6074  
6078  
6082  
6086  
6090  
6094  
6098  
6102  
6106  
6110  
6114  
6118  
6122  
6126  
6130  
6134  
6138  
6142  
6146  
6150  
6154  
6158  
6162  
6166  
6170  
6174  
6178  
6182  
6186  
6190  
6194  
6198  
6202  
6206  
6210  
6214  
6218  
6222  
6226  
6230  
6234  
6238  
6242  
6246  
6250  
6254  
6258  
6262  
6266  
6270  
6274  
6278  
6282  
6286  
6290  
6294  
6298  
6302  
6306  
6310  
6314  
6318  
6322  
6326  
6330  
6334  
6338  
6342  
6346  
6350  
6354  
6358  
6362  
6366  
6370  
6374  
6378  
6382  
6386  
6390  
6394  
6398  
6402  
6406  
6410  
6414  
6418  
6422  
6426  
6430  
6434  
6438  
6442  
6446  
6450  
6454  
6458  
6462  
6466  
6470  
6474  
6478  
6482  
6486  
6490  
6494  
6498  
6502  
6506  
6510  
6514  
6518  
6522  
6526  
6530  
6534  
6538  
6542  
6546  
6550  
6554  
6558  
6562  
6566  
6570  
6574  
6578  
6582  
6586  
6590  
6594  
6598  
6602  
6606  
6610  
6614  
6618  
6622  
6626  
6630  
6634  
6638  
6642  
6646  
6650  
6654  
6658  
6662  
6666  
6670  
6674  
6678  
6682  
6686  
6690  
6694  
6698  
6702  
6706  
6710  
6714  
6718  
6722  
6726  
6730  
6734  
6738  
6742  
6746  
6750  
6754  
6758  
6762  
6766  
6770  
6774  
6778  
6782  
6786  
6790  
6794  
6798  
6802  
6806  
6810  
6814  
6818  
6822  
6826  
6830  
6834  
6838  
6842  
6846  
6850  
6854  
6858  
6862  
6866  
6870  
6874  
6878  
6882  
6886  
6890  
6894  
6898  
6902  
6906  
6910  
6914  
6918  
6922  
6926  
6930  
6934  
6938  
6942  
6946  
6950  
6954  
6958  
6962  
6966  
6970  
6974  
6978  
6982  
6986  
6990  
6994  
6998  
7002  
7006  
7010  
7014  
7018  
7022  
7026  
7030  
7034  
7038  
7042  
7046  
7050  
7054  
7058  
7062  
7066  
7070  
7074  
7078  
7082  
7086  
7090  
7094  
7098  
7102  
7106  
7110  
7114  
7118  
7122  
7126  
7130  
7134  
7138  
7142  
7146  
7150  
7154  
7158  
7162  
7166  
7170  
7174  
7178  
7182  
7186  
7190  
7194  
7198  
7202  
7206  
7210  
7214  
7218  
7222  
7226  
7230  
7234  
7238  
7242  
7246  
7250  
7254  
7258  
7262  
7266  
7270  
7274  
7278  
7282  
7286  
7290  
7294  
7298  
7302  
7306  
7310  
7314  
7318  
7322  
7326  
7330  
7334  
7338  
7342  
7346  
7350  
7354  
7358  
7362  
7366  
7370  
7374  
7378  
7382  
7386  
7390  
7394  
7398  
7402  
7406  
7410  
7414  
7418  
7422  
7426  
7430  
7434  
7438  
7442  
7446  
7450  
7454  
7458  
7462  
7466  
7470  
7474  
7478  
7482  
7486  
7490  
7494  
7498  
7502  
7506  
7510  
7514  
7518  
7522  
7526  
7530  
7534  
7538  
7542  
7546  
7550  
7554  
7558  
7562  
7566  
7570  
7574  
7578  
7582  
7586  
7590  
7594  
7598  
7602  
7606  
7610  
7614  
7618  
7622  
7626  
7630  
7634  
7638  
7642  
7646  
7650  
7654  
7658  
7662  
7666  
7670  
7674  
7678  
7682  
7686  
7690  
7694  
7698  
7702  
7706  
7710  
7714  
7718  
7722  
7726  
7730  
7734  
7738  
7742  
7746  
7750  
7754  
7758  
7762  
7766  
7770  
7774  
7778  
7782  
7786  
7790  
7794  
7798  
7802  
7806  
7810  
7814  
7818  
7822  
7826  
7830  
7834  
7838  
7842  
7846  
7850  
7854  
7858  
7862  
7866  
7870  
7874  
7878  
7882  
7886  
7890  
7894  
7898  
7902  
7906  
7910  
7914  
7918  
7922  
7926  
7930  
7934  
7938  
7942  
7946  
7950  
7954  
7958  
7962  
7966  
7970  
7974  
7978  
7982  
7986  
7990  
7994  
7998  
8002  
8006  
8010  
8014  
8018  
8022  
8026  
8030  
8034  
8038  
8042  
8046  
8050  
8054  
8058  
8062  
8066  
8070  
8074  
8078  
8082  
8086  
8090  
8094  
8098  
8102  
8106  
8110  
8114  
8118  
8122  
8126  
8130  
8134  
8138  
8142  
8146  
8150  
8154  
8158  
8162  
8166  
8170  
8174  
8178  
8182  
8186  
8190  
8194  
8198  
8202  
8206  
8210  
8214  
8218  
8222  
8226  
8230  
8234  
8238  
8242  
8246  
8250  
8254  
8258  
8262  
8266  
8270  
8274  
8278  
8282  
8286  
8290  
8294  
8298  
8302  
8306  
8310  
8314  
8318  
8322  
8326  
8330  
8334  
8338  
8342  
8346  
8350  
8354  
8358  
8362  
8366  
8370  
8374  
8378  
8382  
8386  
8390  
8394  
8398  
8402  
8406  
8410  
8414  
8418  
8422  
8426  
8430  
8434  
8438  
8442  
8446  
8450  
8454  
8458  
8462  
8466  
8470  
8474  
8478  
8482  
8486  
8490  
8494  
8498  
8502  
8506  
8510  
8514  
8518  
8522  
8526  
8530  
8534  
8538  
8542  
8546  
8550  
8554  
8558  
8562  
8566  
8570  
8574  
8578  
8582  
8586  
8590  
8594  
8598  
8602  
8606  
8610  
8614  
8618  
8622  
8626  
8630  
8634  
8638  
8642  
8646  
8650  
8654  
8658  
8662  
8666  
8670  
8674  
8678  
8682  
8686  
8690  
8694  
8698  
8702  
8706  
8710  
8714  
8718  
8722  
8726  
8730  
8734  
8738  
8742  
8746  
8750  
8754  
8758  
8762  
8766  
8770  
8774  
8778  
8782  
8786  
8790  
8794  
8798  
8802  
8806  
8810  
8814  
8818  
8822  
8826  
8830  
8834  
8838  
8842  
8846  
8850  
8854  
8858  
8862  
8866  
8870  
8874  
8878  
8882  
8886  
8890  
8894  
8898  
8902  
8906  
8910  
8914  
8918  
8922  
8926  
8930  
8934  
8938  
8942  
8946  
8950  
8954  
8958  
8962  
8966  
8970  
8974  
8978  
8982  
8986  
8990  
8994  
8998  
9002  
9006  
9010  
9014  
9018  
9022  
9026  
9030  
9034  
9038  
9042  
9046  
9050  
9054  
9058  
9062  
9066  
9070

In Steps 1 through 3 in the figure, when class file data have been created for a designated class, various reference tables, including class, field and method tables, are set up just as in the prior art.

5 In Steps 4 through 6, the aforesaid resolution unit 5 resolves the references using the variable and method names in the class file data. Until references have been resolved for all the class file data, classes whose references are still unresolved will be designated in order in Step 6, and their references will be resolved. When references have been resolved with respect to the class file data for every class, the answer in Step 5 will go to "no", and instructions will be executed in order in the loop in Steps 7 and 8.

10  
15 Figure 4 shows details of the processing in Step 4 of the aforesaid Figure 3. To simplify the explanation, the order shown is that used to obtain a field table index by resolving a field reference.

We shall next explain, with reference to the flow chart in Figure 4, the order of processing used to resolve the field reference in the class file data shown in the aforesaid Figure 2.

20 In Step 4-1, resolution unit 5 extracts from the class file data an entry (entry number 6) whose references have not yet been resolved. This will be an entry with the field reference tag "09" and which does not have a resolved field index. In Step 4-2, unit 5 traces in order, from the aforesaid data it extracted in Step 4-1, the data for the class name reference (entry number 1) and its link information (entry number 20), to obtain the name of the class to which the aforesaid variable belongs.

25 Resolution unit 5 checks whether a class with this name can be found in the class table. If it is there, unit 5 obtains a class table index for that class (Steps 4-3 and 4-5). If there is no class in the table with that name, in Step 4-4 unit 5 uses the aforesaid class file data for that class to enter it in the class table and the field table, and it obtains a class table index.

From the data in entry number 6 it extracted in the aforesaid Step 4-1, resolution unit 5 then follows, in order, the reference data for the name and type (entry number 9) and their link information (entry numbers 27 and 17). In this way it obtains the variable and class names (Step 4-6). It retrieves the field table and gives it the class table index obtained in the 5 aforesaid Step 4-5. It extracts the field data which match the variable and type names obtained in Step 4-6 and obtains the field table index attached to these data (Step 4-8).

When it obtains the field table and class table indices in the aforesaid Steps 4-5 and 4-8, resolution unit 5 writes them into the designated blank space in the aforesaid class file data 10 (Step 4-9).

Unit 5 continues, in the same way, to extract in order the data whose references are not yet resolved, the data which have the field reference tag "09". It obtains field and class table indices from the class, variable and type names it finds by following the data linked to these, and it writes the values for each index in the class file data. 15

When unit 5 retrieves a field table and finds that it has no field data which match the combination of class table index, variable name and type name, the answer in Step 4-7 goes to "no", and we move to Step 4-11, which is processed as a link error. 20

Thus the field references for every program variable are resolved through the variables themselves before the program is executed. The resolved field table index is stored in the blank space within the entry number representing the instruction which indicates that field reference. Thus when execution unit 3 is going to execute the aforesaid instruction 10, it obtains 25 the aforesaid resolved field table index from the entry number written into the operand of instruction 10. It accesses the field it finds in the object offset value it obtains from the values in the index and executes the instruction.

The reference data containing character data representing the name of the class (consisting of entry number 1 with the aforesaid tag "07" attached to it and entry number 20, the 30

linked entry number which represents the character string for the class name) are linked to the instruction which required the class reference. The aforesaid resolved class table index is written into the blank space in the head entry number. Accordingly, execution unit 3 obtains the aforesaid resolved class table index through the entry number written into the operand. By 5 reading out the class data which correspond to this index, it can easily obtain the data associated with the objects and methods in that class.

References are also resolved in the same way before executing the program for instructions requiring that a method be referenced. In this case a resolved method table index is 10 written into the head entry number of the reference data linked to the instruction. When this instruction is to be executed, the method table index obtained from the entry number in the instruction's operand tells unit 3 where the program for the method is stored, and the method can be accessed immediately.

15 If a program contains an instruction which generates a string object representing a character string in the class, the references are resolved for that object's name the first time the instruction is executed. The results are written into the class file data so that instructions which require that the string object be referenced can be executed at high speed.

20 Two tables are set up for a string object: a character string table in which is stored a character string representing that object and an object table which indicates the class to which the object belongs and the location in the aforesaid character string table where its character string is stored. When an instance is generated, data are entered which are linked mutually and reciprocally to the aforesaid character string and object tables. When a reference is resolved, 25 just as was described above, an object table index obtained by resolving the references is written into the head entry number of the reference data linked to the instruction. Thereafter, whenever a program requires that a string object be referenced, the object table index for the object indicated by the class file data is read out. The location in the character string table where the string is stored can be found quickly so that the pertinent character string data can 30 be accessed.

With the information processing system described above, when name data are referenced, all the reference data are extracted from the class file and the reference processing is resolved before the instruction indicating that a given address should be accessed is carried out.

5 Since the results are stored in the location where read-out of the reference data linked to the instruction's operand begins, the addresses of the destinations to be accessed for each instruction can be completely specified using the results in the class file data as soon as the program is started up, and the instructions can be carried out at high speed. There will be no difference between the program's running times the first time it is executed and the subsequent times it

10 is executed. This allows the program's running time to be predicted accurately.

Since the results of referencing are written into the aforesaid head entry of the operand without rewriting the instruction, the results are linked to the program using the existing link configuration. To execute an instruction using these referencing results, execution unit 3 has only to read the results of referencing out of the head entry written in the operand. There is no need to significantly change the design of the interpreter.

Because the referencing results are written in a specific location in the head entry, they can be read out easily. When resolving the references, the processing involved in finding data whose references are still unresolved is also simplified.

In the information processing system of this embodiment, there is no need to rewrite instructions. Reading an instruction out of the ROM and executing it doesn't require that the program be loaded at start-up time. Once power is supplied to the device, the system can start up swiftly. Because it is not necessary to load the program, the capacity of the RAM can be substantially reduced.

Because the aforesaid results of referencing are stored so as to correspond to the reference data in the class file, instructions which require that the same data be referenced can share

the results. There is thus no need to resolve the same reference repeatedly, and the references can be resolved more efficiently.

5 To apply the method of storing the referencing results for each set of reference data in the aforesaid class file data, the class file data in which are written the referencing results already obtained can be stored in the ROM, or a reference table can be created according to the class file data and both the data and the table can be stored in the ROM. With this method, it will not be necessary to resolve the references after starting up the system. As soon as the program is activated, high-speed processing can commence. This information processing system is 10 particularly suited for use in compact devices which require real-time processing.

Figure 5 shows the configuration of an information processing system related to the second preferred embodiment of this invention.

15 The information processing system of this embodiment has the same configuration as that shown in Figure 1 above, with the exception of resolution unit 5. However, all the reference tables created by storage unit 2 and table management unit 4 are set up in the ROM along with instruction storage unit 1. The rest of the configuration is identical to that of the aforesaid first embodiment.

20 Class file data for all classes stored in instruction storage unit 1 are stored in storage unit 2 for class file data. Table management unit 4 manages the class, field, method and character string tables in which class data are recorded.

25 Index data representing the results already obtained by resolving the references for the variable, method and class name for each class are stored in storage unit 2 for class file data, just as in the aforesaid first embodiment.

30 With an information processing system configured as described above, the system is activated as soon as power is connected. Class file data in which the referencing results are writ-

ten and reference tables are set up, and all instructions are carried out at high speed based on the results of referencing those instructions. This information processing system does not require a large-capacity RAM, so the device which contains it can perform high-speed information processing at a low cost.

5

With the inventions disclosed in Claims 1 and 2 of this application, the reference processing is resolved before the program is executed using the reference data which specify the locations in the memory to be accessed. The results of referencing are then linked to the program through the reference data. Thus when the program is started up, the stored referencing results can be used to access the memory quickly. This scheme stabilizes the program's running time and allows it to run at high speed. Because the program's running time is invariant, the information processing system is suitable for a device using real-time processing, which requires an accurate prediction of the program's running time.

10  
11  
12  
13  
14  
15

There is no need to rewrite the program when the references are resolved. This obviates the need to load the program into a RAM. Since less time is required from the moment the power is connected to the moment the system is activated, the capacity of the RAM can be significantly reduced. References need to be resolved only once for programs which indicate that the same data be referenced more than once. References can be resolved in a shorter time, and the program will be executed sooner.

20

With the invention in Claim 3, the existing configuration of the program's links is used to link the referencing results to the program. There is thus no need to greatly modify the design of the interpreter.

25

With the invention disclosed in Claim 4, the referencing results can be obtained efficiently when the program is to be run from a designated location in the head code data of the link information. This allows the program to run even faster. When the references have been resolved before the program is executed, the program checks whether the results have been

written into the aforesaid designated location. This allows data with unresolved references to be recognized easily so that resolution processing can be speeded up.

With the invention disclosed in Claim 5 of this application, the program to be executed and the referencing results linked to it are read out of the ROM before the program is run. Then as soon as power is connected, the program can be activated and high-speed processing can be performed. There is no need to load the program or the results into the RAM, so the RAM's capacity can be further reduced. This scheme allows a device to have a program in it which can execute high-speed processing at a low cost.